UNITED STATES PATENT AND TRADEMARK OFFICE

[illegible]

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/656,005	09/05/2003	Gary K. Law	06005/39537	8163

4743 7590 06/14/2007
MARSHALL, GERSTEIN & BORUN LLP
233 S. WACKER DRIVE, SUITE 6300
SEARS TOWER
CHICAGO, IL 60606

EXAMINER	
THERIAULT, STEVEN B	
ART UNIT	**PAPER NUMBER**
2179	
MAIL DATE	**DELIVERY MODE**
06/14/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 10/656,005	**Applicant(s)** LAW ET AL.
	Examiner Steven B. Theriault	**Art Unit** 2179

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 March 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-79* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-79* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the following communications: Amendment filed 03/19/2007.

This action is made Final.

2. Claims 1 –79 are pending in the case. Claims 1, 18, 34, and 58 are the independent claims. Claim 58 is the amended claim. The applicant is advised that a new examiner has been assigned to the case.

Claim Rejections - 35 USC § 102

3. **The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:**

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. **Claims 1, 15 – 18, 31 – 46, 48 – 50, and 53 – 55 are rejected under 35 U.S.C. 102(b) as being anticipated by Kodosky et al. (Patent No. 6,954,724).**

As to claims 1 and 18, Kodosky et al. teaches:

A computer implemented method and corresponding apparatus for configuring, via a computing device (computer 102 – see e.g., Fig. 2 and col. 7, lines 19 – 26) having a display device (video display subsystem 180 – see e.g., Fig. 2 and col. 10, lines 1 – 6) and an input device (keyboard – see e.g., col. 7, lines 19 – 26), a function block (function block – see e.g., col. 11, lines 59 – 67) associated with a (industrial automation – see e.g., col. 2, lines 8 - 15), the function block (function block – see e.g., col. 19, lines 49 – 59) to implement a state machine (state machine – see e.g., col. 19, lines 49 – 59; i.e., the while loop function block is used to implement the transition of the state machine), the method comprising the steps/means for: providing a graphical user interface (graphical program – see e.g., col. 2, lines 58 – 67) via the display device for configuring (display screen – see e.g., col. 7, lines 19 – 26), at least in part, how the state machine is to transition among a plurality of states (see e.g., Fig. 16 and col. 48 – 67; i.e., the while loop function block controls the state machines transition of states), wherein the graphical user interface includes a plurality of graphical elements (see e.g., Fig. 16 and col. 20, lines 44; i.e., the graphical elements corresponds to "Adder 532", "AND gate 534" etc.), wherein at least some of the graphical elements can be used to indicate desired transitions between states (see e.g., Fig. 16 and col. 20, lines 30 – 59; i.e., "Adder 532", "AND gate 534", etc. are graphical elements used to indicate the transition between states); wherein the at least one input is to be associated with the process plant (see e.g., col. 8, lines 5 – 18); receiving state transition data via the graphical user interface (see e.g., col. 13, lines 40 – 54); and storing the state transition data on a first computer readable medium associated with the function block (see e.g., col. 13, lines 40 – 54; i.e., the arrangement of graphical elements on the screen by the user is stored by the graphical programming system).

As to claims 15, Kodosky et al teaches:

A method according to claim 1, wherein the at least one input (see e.g., col. 15, lines 54 – 67; i.e., the AND-gate must have proper input to enable itself within the graphical program) is to be associated with at least one of a process control system (industrial automation system 140 – see e.g., col. 8, line 5 – 18; i.e., the industrial automation system 140 is used to construct the input/output of function blocks), a simulation of a process control system (see e.g., col. 7, lines 25 – 26; i.e., analyzing the graphical program is a form of simulation), a safety system, and a simulation of a safety system.

As to claim 16, Kodosky et al. teaches:

A method according to claim 1, wherein the at least one input (see e.g., Fig. 19 and col. 12, lines 1 – 11; i.e., at least one output, indicated by graphical connectors, are connected to a corresponding function block as an input) is to be received from at least one other function block (function block – see e.g., Fig. 19 and col. 11, lines 59 – 67; i.e., the output of a function blocks or logical gate corresponds to an input of a corresponding function block or logical gate) associated with the process plant.

As to claim 17, Kodosky et al. teaches:

A method according to claim 1, wherein the at least one input (see e.g., col. 13, lines 30 – 48; i.e., the input corresponds to the user manipulating and connecting various icons on the display screen) is to be received from an operator interface (keyboard – see e.g., col. 7, lines 19 – 26).

As to claim 31, claim 31 differs from claim 15 only in that claim 31 is an apparatus claim using a tangible medium (see e.g., col. 8, lines 59 – 67) containing machine readable instructions (see e.g., col. 8, lines 59 – 67) when executed causes a processor (see e.g., col. 9, lines 1 – 6) to perform the steps of claim 15. Thus, claim 31 is analyzed as previously discussed with respect to claim 15 above.

As to claim 32, claim 32 differs from claim 16 only in that claim 32 is an apparatus claim using a tangible medium (see e.g., col. 8, lines 59 – 67) containing machine readable instructions (see e.g., col. 8, lines 59 – 67) when executed causes a processor (see e.g., col. 9, lines 1 – 6) to perform the steps of claim 16. Thus, claim 32 is analyzed as previously discussed with respect to claim 16 above.

As to claim 33, claim 33 differs from claim 17 only in that claim 33 is an apparatus claim using a tangible medium (see e.g., col. 8, lines 59 – 67) containing machine readable instructions (see e.g., col. 8, lines 59 – 67) when executed causes a processor (see e.g., col. 9, lines 1 – 6) to perform the steps of claim 17. Thus, claim 33 is analyzed as previously discussed with respect to claim 17 above.

As to claim 34, Kodosky et al. teaches:

A method of implementing a function block (function block – see e.g., col.3, lines 55 – 58) for use in controlling, or simulating control of (see e.g., col. 7, lines 24 – 26; i.e., computer 102 is used to control and analyze a unit under test), one or more field devices (see e.g., col. 5, lines 22 – 27; i.e., the target device corresponds to a field device) in a process plant (industrial automation – see e.g., col.2, lines 8 - 15), the method comprising: providing a graphical user interface (graphical program – see e.g., col. 2, lines 58 – 67) via the display device (video display subsystem 180 – see e.g., Fig. 2 and col. 10, lines 1 – 6) for configuring (see e.g., col. 13, lines 25 – 28; i.e., the user places and connects various icons on the display screen), at least in part (see e.g., col. 8 , lines 57 – 61; i.e., portion of a graphical program), how the state machine is to transition among a plurality of states (see e.g., col. 19, lines 48 – 63; i.e., the state machines within the while transitions from state A through state Done), wherein the graphical user interface includes a plurality of graphical elements (see e.g., Fig. 16 and col. 20, lines 44; i.e., the graphical elements corresponds to "Adder 532", "AND gate 534" etc.), wherein at least some of the graphical elements can be used to indicate desired transitions between states (see e.g., Fig. 19 and col. 22, lines 17 – 20; i.e., the diagram signal is enabled to start the transition within the loop); wherein the at least one input is to be indicative of conditions within the process plant (see e.g., Fig. 16 and col. 13, lines 31 – 34); receiving state transition data via the graphical user interface (see e.g., col. 21, lines 2 – 6); storing the state transition data on a first computer readable medium associated with the function block (see e.g., col. 13, lines 40 – 54; i.e., the arrangement of graphical elements on the screen by the user is stored by the graphical programming system); receiving the at least one input, wherein the at least one input is associated with the process plant (see e.g., Fig. 17; i.e., the zero constant corresponds to an input and the graphical diagram is associated with the process plant); determining a next state based (see e.g., col. 14, lines 21 – 26; i.e., the next state corresponds to a signal being generated then a node completes and the output is used to enable subsequent nodes), at least in part, on at least one of the at least one input (see e.g., col. 14, lines 14 – 17; i.e., the input for each node includes an "Enable input", "Clear_Enable signal input", "clock signal input", and an "Enable_Out signal input"), a current state (see e.g., col. 20, lines 17 – 19; i.e., after the traversal of the loop, the current state is initialized as the Done signal), and the state transition data stored on a first computer readable medium (see e.g., col. 11, lines 25 – 30; i.e., memory 214 is used to store FPGA state information); setting the current state of the state machine to the next state (see e.g., col. 14, lines 21 – 24; i.e., the current state of a node is used to activate the next state of a subsequent node); and providing at least one function block output (see e.g., Fig. 14 and col. 6, lines 6 – 7; i.e., each node corresponds to a function block that has an output, which results from successful transversal) for use in controlling the one or more field devices to at least a second other function block (see e.g., col. 19, lines 49 – 59), wherein the at least one function block output is based on the current state of the state machine (see e.g., col. 19, lines 49 – 59).

As to claim 35, Kodosky et al. teaches:

A method according to claim 34, wherein the at least one input comprises a plurality of inputs (see e.g., col. 14, lines 17 – 19; i.e., the Enable inputs comprises other inputs in order to execute at a proper time); wherein determining the next state comprises determining the next state further based on priorities associated with the plurality of inputs (see e.g., col. 20, lines 4 – 10; i.e., the priority corresponds to the sequential transition from node A through node Done, wherein node Done corresponds to the next state of the state machine).

As to dependent claim 36, Kodosky et al teaches:

A method according to claim 35, wherein determining the next state (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., to advance to the next state or node, the input must satisfy the nodes condition) further based on priorities (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., the priority is illustrated as alphanumeric letters, wherein node A is evaluated before node B) associated with the plurality of inputs comprises determining the next state further based on an order associated with the plurality of inputs (see e.g., Fig. 14; i.e., the plurality of inputs are specified by arrows, wherein the next state depends on the input received from previous nodes).

As to dependent claim 37, Kodosky et al. teaches:

A method according to claim 34, further comprising: determining whether a state transition is to occur based on at least one of the at least one input (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., a state transition of a corresponding node will occur if the input satisfies the nodes condition, such as the output of node A as the input of node B) and the state transition data stored on the first computer readable medium (see e.g., col. 11, lines 25 – 30; i.e., memory 214 is used to store FPGA state information); wherein determining the next state comprises determining the next state if a state transition is to occur (see e.g., col. 19, lines 48 – 59; i.e., the loop determines the next state of a state transition); and wherein setting the current state of the state machine to the next state comprises setting the current state of the state machine to the next state if a state transition is to occur (see e.g., col. 20, lines 4 – 14; i.e., when node Done is reached, the current state of the state machine is set to the next state).

As to dependent claim 38, Kodosky et al. teaches:

A method according to claim 34, wherein determining the next state (see e.g., Fig. 14 and col. 19, lines 53 – 54; i.e., the determination of the next state is determined by the Loop Enable signal) comprises determining one or more, if any, of the at least one input that is a particular value (see e.g., Fig. 14 and col. 19, lines 53 - 54; i.e., the particular value corresponds the Loop Enable signal, wherein the state machine remains in state B until the signal is "true").

As to dependent claim 39, Kodosky et al teaches:

A method according to claim 38, wherein determining the next state (see e.g., Fig. 14 and col. 19, lines 53 – 54; i.e., the determination of the next state is determined by the Loop Enable signal) further comprises determining one or more, if any, of the one or more of the at least one input that are the particular value (see e.g., Fig. 14 and col. 19, lines 53 - 54; i.e., the particular value corresponds the Loop Enable signal, wherein the state machine remains in state B until the signal is "true") and that also correspond to state changes based on the state transition data (see e.g., col. 11, lines 25 – 30; i.e., the state information corresponds to the state transition data) stored on the first computer readable medium (see e.g., col. 11, lines 25 – 30; i.e., memory 214 is used to store FPGA state information).

As to claim 40, Kodosky et al. teaches:

A method according to claim 39, further comprising selecting (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., the selection is made from each structured node function block in order to generate an input for the next node) one of the one or more, if any, of the at least one input (see e.g., Fig. 13 and col. 19, lines 36 – 47; i.e., the function block for a structured node corresponds to Fig. 14, wherein the function block for a specific node comprises multiple inputs, such as "Enable In input", "Period input", "Phase input", and "Loop Control input") that are the particular value (see e.g., Fig. 14 and col. 19, lines 53 - 54; i.e., the particular value corresponds the Loop Enable signal, wherein the state machine remains in state B until the signal is "true") and that correspond to state changes (see e.g., col. 19, lines 48 – 55; i.e., the inputs used corresponds to signals for state changes).

As to dependent claim 41, Kodosky et al. teaches:

A method according to claim 40, wherein the at least one input (see e.g., col. 15, lines 55 – 60; i.e., the one input corresponds to the AND-gate) comprises a plurality of inputs (see e.g., col. 15, lines 55 – 60; i.e., a basic node declares an AND-gate, in which the AND-gate has a plurality of inputs); wherein selecting one of the one or more, if any, of the at least one input that are the particular value (see e.g., col. 17, lines 39 – 50; i.e., the AND-gate is used to determine whether the inputs that are a particular value are selected and received) and that correspond to state changes (see e.g., col. 17, lines 39 – 50; i.e., the state change corresponds to the function receiving an output signal from an AND-gate) comprises selecting based on priorities associated with the plurality of inputs (see e.g., Fig. 14; i.e., priorities are alphanumeric based).

As to dependent claim 42, Kodosky et al teaches:

A method according to claim 41, wherein selecting one of the one or more, if any, of the at least one input that are the particular value (see e.g., col. 17, lines 39 – 50; i.e., the AND-gate is used to determine whether the inputs that are a particular value are selected and received) and that correspond to state changes (see e.g., col. 17, lines 39 – 50; i.e., the state change corresponds to the function receiving an output signal from an AND-gate) comprises selecting based on an order (see e.g., Fig. 14; i.e., the alphanumeric nodes corresponds to the order and each node has a plurality of inputs) associated with the plurality of inputs (see e.g., Fig. 14 and col. 17, lines 39 – 50; i.e., order are alphanumeric based, wherein each node has a plurality of inputs).

As to dependent claim 43, Kodosky et al. teaches:

A method according to claim 34, wherein determining the next state comprises determining one or more, if any, of the at least one input (see e.g., col. 20, lines 8 – 14; i.e., the determination of at least one input corresponds to the inputs from one node to another of Fig. 14) associated with potential state changes from the current state (see e.g., col. 20, lines 8 – 14; i.e., the state machine traverses from state A and waits at state E until the period is done, which then advances to state Done) based on the state transition data stored on the first computer readable medium (see e.g., col. 11, lines 25 – 30; i.e., memory 214 is used to store FPGA state information).

As to dependent claim 44, Kodosky et al teaches:

A method according to claim 43, wherein determining the next state further comprises determining one or more, if any, of the one or more of the at least one input (see e.g., col. 20, lines 8 – 14; i.e., the determination of at least one input corresponds to the inputs from one node to another of Fig. 14) associated with potential state changes from the current state (see e.g., col. 20, lines 8 – 14; i.e., the state machine traverses from state A and waits at state E until the period is done, which then advances to state Done) and that also are a particular value (see e.g., col. 20, lines 8 – 14; i.e., the particular value corresponds to the output of state Done, wherein the value of state E must be a particular value to proceed to state Done).

As to dependent claim 45, Kodosky wt al. teaches:

A method according to claim 34, wherein providing the at least one function block output (see e.g., col. 19, lines 40 – 41; i.e., the While loop function block provides an index output) comprises providing a plurality of function block outputs (see e.g., col. 19, lines 44 – 47; i.e., the While loop function bock further outputs other output signals, such as, Clear and Enable output signals).

As to dependent claim 46, Kodosky et al. teaches:

A method according to claim 45, wherein each of at least some of the plurality of function block outputs (see e.g., Fig. 14 and col. 19, lines 49 – 60) are indicative of whether the current state of the state machine corresponds to a respective one of a plurality of possible states of the state machine (see e.g., col. 20, lines 8 – 19; i.e., if the loop state of the state machine is still enabled as the current state, the state machine advances back to state C).

As to dependent claim 48, Kodosky et al. teaches:

A method according to claim 45, wherein providing the plurality of function block outputs (see e.g., Fig. 14 and col. 14, lines 44 – 47; the function block for a structured node has Clear and Enable outputs) comprises providing one function block output indicative of the current state of the state machine (see e.g., Fig. 14; i.e., the output of the Done node corresponds to the output of the current state of the state machine).

As to dependent claim 49, Kodosky et al. teaches:

A method according to claim 34, wherein the at least one function block output comprises one function block output indicative of the current state of the state machine (see e.g., Fig. 14 and col. 20, lines 4 – 7; i.e., the structured function block node Done indicates the state machine has completed the loop, wherein the output of node Done corresponds to the current state of the state machine).

As to dependent claim 50, Kodosky et al. teaches:

A method according to claim 34, further comprising: receiving an input indicative of whether the state machine function block is to be disabled (see e.g., col. 19, lines 66 – 67; i.e., the error input corresponds to indicating whether the state machine function block is to be disabled); and if the input indicative of whether the state machine function block is to be disabled indicates that the state machine function block is to be disabled, setting the current state of the state machine to a disabled state (see e.g., Fig. 14 and col. 19, lines 66 – 67; i.e., once state Error is reached, the loop completes, therefore the current state of the state machine is set to a disabled state).

As to dependent claim 53, Kodosky et al. teaches:

A method according to claim 34, wherein receiving the at least one data input (see e.g., col. 19, lines 37 – 40; i.e., the inputs correspond to period, phase, and control inputs) to the state machine function block (see e.g., col. 16, line 36) comprises receiving at least one signal (see e.g., col. 19, lines 37 – 40) associated with at least one of a process control system (process control application – see e.g., col. 8, line 3), a simulation of a process control system (see e.g., col. 8, lines 1 – 3; i.e., data acquisition for test and measurement applications correspond to simulation of a process control system), a safety system, and a simulation of a safety system.

As to dependent claim 54, Kodosky et al. teaches:

A method according to claim 34, wherein the at least one input (see e.g., Fig. 14 and col. 19, lines 37 – 40; i.e., the inputs corresponds to enabling and phase inputs) is to be received from at least one other function block (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., the function block corresponds to node A through node Done) associated with the process plant (see e.g., col. 8, lines 5 – 18).

As to dependent claim 55, Kodosky et al. teaches:

A method according to claim 34, wherein the at least one input (see e.g., col. 13, lines 31 – 34) is to be received from an operator interface (keyboard – see e.g., col. 7, lines 19 – 26).

Claim Rejections - 35 USC § 103

5. **The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:**

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. **Claims 2 – 5, 9, 19 – 22, 26, 47, 58 – 74, 76, and 78 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kodosky et al. (Patent No. 6,954,724) in view of Klapper et al. (Pub No. 2002/0194218).**

As to dependent claims 2 and 19, Kodosky et al. teaches a computer implemented method and corresponding apparatus, wherein the steps/means for providing a plurality of graphical elements (see e.g., Fig. 16 and col. 20, lines 44; i.e., the graphical elements corresponds to "Adder 532", "AND gate 534" etc.) associated with function blocks (function block – see e.g., col. 11, lines 59 – 67; i.e., the function blocks and logical gates corresponding to graphical elements are associated through the graphical program), and state machines (state machine – see e.g., col. 19, lines 49 – 59), wherein at least some possible pairings of one of the at least one input and one of the states of the plurality of states of the state machine (see e.g., col. 2, lines 58 – 67 and col. 3, lines 1 – 17; i.e., the user utilizes the graphical programming environment to place icons on the diagram editor to manipulate and find possible paring of logical gates, and function blocks described in Fig. 19). Kodosky et al. further teaches receiving state transition data (see e.g., col. 13, lines 40 – 54; i.e., the arrangement of graphical elements on the screen by the user is stored by the graphical programming system) by receiving data (see e.g., col. 2, lines 57 – 67; i.e., the data corresponds to the manipulation of icons used in the graphical programming environment by the user) via an input device (keyboard – see e.g., col. 7, lines 19 – 26), and the state is indicative of a next state to which the state machine should transition (see e.g., Fig. 14 and col. 19, lines 48 – 67; i.e., the while loop function block controls the next state, in which transition of the state machine corresponds to traversing through "state A" through "state Done"), but does not teach cells associated with possible parings, via an input device, of graphical elements with at least one input that is indicative of a next state. Klapper et al. teaches a "Cause and Effects Matrix" for a process control system, in which the user has the ability to define the relationship between a particular cause and a particular effect, via an input device (see e.g., para. [0029]), at the intersection of a Cause row and an Effect column (see e.g., para. [0027]; i.e., the user manipulates the "Cause and Effects Matrix" in order to discover a next state transition by possible pairings of Cause rows and Effect columns). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the graphical user interface for configuring function blocks and state machines of Kodosky et al. with the "Cause and Effect Matrix" of Klapper et al. because the user does not have to be concerned with testing and retesting the logic when the logic for a matrix is initially generated or modified (see e.g., para. [0027]).

As to dependent claims 3 and 20, Kodosky et al. teaches a computer implemented method and corresponding apparatus, wherein the steps/means for displaying (see e.g., col. 2, lines 58 – 67) on the display device (video display subsystem 180 – see e.g., Fig. 2 and col. 10, lines 1 – 6) indications of state transition (see e.g., Fig. 19; i.e., the graphical programming environment can indicate state transition by graphical connectors), but does not teach displaying on a display device indications of the state transition data in appropriate cells of the plurality of cells. Klapper et al. teaches a "Cause and Effect Matrix" that allows the user to define sensors, trip definitions, actuators, output definitions, etc. and display the indication of state transition in appropriate cells (see e.g., Fig. 1 and para. [0027]; i.e., the indication of state transition can be viewed through the intersection of Cause row and Effect column). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the graphical user interface for configuring function blocks and state machines of Kodosky et al. with the "Cause and Effect Matrix" of Klapper et al. because the user does not have to be concerned with testing and retesting the logic when the logic for a matrix is initially generated or modified (see e.g., para. [0027]).

As to dependent claim 4 and 21, Kodosky et al. teaches a computer implemented method and corresponding apparatus, wherein the steps/means for displaying (see e.g., col. 2, lines 58 – 67) on a display device (video display subsystem 180 – see e.g., Fig. 2 and col. 10, lines 1 – 6), state transition of function blocks and state machines (see e.g., col. 19, lines 48 – 67) through a graphical user interface (graphical program – see e.g., col. 2, lines 58 – 67), but does not teach displaying on a display device a plurality of cells within a matrix, in which the matrix comprises at least one row of cells and a plurality of column of cells, wherein at least one row is associated with a corresponding input, and wherein each column is associated with a state. Klapper et al. teaches a "Cause and Effect Matrix" (see e.g., para. [0027]) with a plurality of cells (see e.g., Fig. 1) in which the matrix comprises at least one row (see e.g., Fig. 1 and para. [0027]) that is associated with an input (see e.g., para. [0029]) and a plurality of columns (see e.g., Fig. 1 and para. [0027]) that is associated with a state (see e.g., Fig. 1 and para. [0032]; i.e., the state transition corresponds to the output of the intersection of row and column). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the display device, and state transition of function blocks and state machines through a graphical user interface of Kodosky et al. with the "Cause and Effect Matrix" of Klapper et al. because the intersection functional unit includes predefined, configurable logic for defining the relationship between the input parameter and the output response (see e.g., para. [0008]).

As to dependent claim 5 and 22, Kodosky et al. teaches a computer implemented method and corresponding apparatus, wherein the steps/means for displaying (see e.g., col. 2, lines 58 – 67) on a display device (video display subsystem 180 – see e.g., Fig. 2 and col. 10, lines 1 – 6), state transition of function blocks and state machines (see e.g., col. 19, lines 48 – 67) through a graphical user interface (graphical program – see e.g., col. 2, lines 58 – 67), but does not teach does not teach displaying on a display device a plurality of cells within a matrix, in which the matrix comprises a plurality of rows of cells and at least one column of cells, wherein the plurality of rows is associated with a corresponding state of the plurality of states, and at least one column is associated with a corresponding input. Klapper et al. teaches a "Cause and Effect Matrix" (see e.g., para. [0027]) with a plurality of cells (see e.g., Fig. 1) in which the matrix comprises a plurality of rows of cells (see e.g., Fig. 1) and at least one column of cells (see e.g., Fig. 1), wherein the rows are associated with a plurality of states (see e.g., para. [0029] – [0031]; i.e., the states corresponds to the condition that will occur if two out of the three sensors satisfy a set of threshold conditions) and at least one column is associated with an input (see e.g., para. [0034]; the input corresponds to the selection of the "Latched" option, which is actuated by the operator when depressing the restart button). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the display device, and state transition of function blocks and state machines through a graphical user interface of Kodosky et al. with the "Cause and Effect Matrix" of Klapper et al. because by defining the cause parameter, effect parameter, and intersection parameter, the user creates a matrix database that holds all the information defining the constructed matrix (see e.g., para. [0037]).

As to dependent claim 9 and 26, Kodosky et al. teaches a computer implemented method and corresponding apparatus, wherein the steps/means for receiving data (Fig. 19 and col. 2, lines 58 – 67; i.e., the user manipulates icons in a diagram editor to create a data flow program), via an input device (see e.g., Fig. 2 and col. 7, lines 19 – 26), indicative of a number of states in the plurality of states (see e.g., Fig. 19; i.e., the logical gates and function blocks can indicate the number of states through "wire up", which are displayed as connectors from one icon to another, on the graphical editor), but does not teach determining a number of cells based on the number of states. Klapper et al. teaches an interface that allows a user to define sensors, trip definitions, actuators, and output definitions (see e.g., para. [0027]; i.e., the sensors, trip definition, actuators, and output definitions corresponds to states) that are directly transferred to a "Cause and Effect Matrix" (see e.g., Fig. 1 and para. [0027]; i.e., the user is able to define the Cause row and Effect column, which is then passed to a matrix functional unit 18, that automatically determines the number of cells based on the number of states). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate receiving data via an input device, and indicating a number of states in the plurality of states of Kodosky et al. with the matrix functional unit of the "Cause and Effect Matrix" of Klapper et al. because by self generating the necessary logic, it removes the need for a control engineer to generate logic for implementing the matrix (see e.g., para. [0027]).

As to dependent claim 47, Kodosky et al. teaches displaying (see e.g., col. 2, lines 58 – 67) on a display device (video display subsystem 180 – see e.g., Fig. 2 and col. 10, lines 1 – 6), state transition of function blocks and state machines (see e.g., col. 19, lines 48 – 67) through a graphical user interface (graphical program – see e.g., col. 2, lines 58 – 67), and providing a plurality of function block outputs (see e.g., Fig. 14 and col. 19, lines 48 – 59), but does not teach retrieving, based on the current state, data indicative of appropriate values for at least some of the plurality of state machine function block outputs from an output configuration database, and setting at least some of the plurality of function block outputs to the appropriate value. Klapper et al. teaches retrieving data indicative of appropriate values for at least some of the plurality of state machine function block outputs (see e.g., para. [0045]; i.e., Effect Latch Function logic element 124, receives a Restart Tag, and a Max Restart Time value 126 from the effect database) from an output configuration database (see e.g., para. [0045]; i.e., the output configuration database corresponds to the effect database), and setting at least some of the plurality of function block outputs to the appropriate value (see e.g., para. [0047]). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the graphical user interface, state machine, function block, and providing a plurality of function block output of Kodosky et al. with the retrieval of function block outputs from a output configuration table, and setting the plurality of function block outputs to the appropriate value of Klapper et al. because the logic that implements the matrix is automatically defined and established by transferring the matrix to the programmable logic controllers (see e.g., para. [0049]).

As to independent claim 58, Kodosky et al. teaches a state transition data (see e.g., Fig. 14 and col. 20, lines 8 – 14; i.e., the state transition data corresponds to the output of the Done state) indicative of how a state machine implemented by the function block (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., the structure node function blocks, such as node A through node Done, are function blocks that implement the transition state of a state machine) is to transition among a plurality of states (see e.g., Fig. 14 and col. 19, lines 48 – 59), wherein the state transition data comprises data for at least some possible pairings (see e.g., col. 21, lines 28 – 26; i.e., the graphical program is used to arrange icons on a graphical editor for possible pairing of elements, and the data structure of Fig. 18 is the data that reflects the paired elements) of the plurality of states of at least one input to the function block (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., the plurality of states corresponds to node A through node Done, in which inputs are used to evaluate each node), which indication of a next state to which the state machine should transition when the state machine is in the corresponding state (see e.g., Fig. 14 and col. 20, lines 8 – 14; i.e., the node and output of node Done corresponds to the next state a state machine should transition) and when the corresponding input is a particular value (see e.g., col. 20, lines 8 – 19; i.e., if node E is a particular value, it will re-execute the loop until a particular value is reached in order to advance to node Done). Kodosky et al. further teaches a computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks), which includes code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program) to receive at least one input (see e.g., col. 13, lines 21 –39; i.e., the inputs corresponds to the user manipulating icons on the graphical editor) that comprises data associated with the process plant (see e.g., col. 8, lines 5 – 18), a second code to determine the next state of the state machine (see e.g., Fig. 14 and col. 20, lines 4 – 14; i.e., the code corresponds to the graphical program stored on a CD-ROM, magnetic media, or floppy disk that executes code to perform the next state determination of the state machine), a third code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program that includes setting the state machine to the next state) to set the current state of the state machine to the next state (see e.g., col. 20, lines 8 – 14; i.e., successful traversal to node Done sets the state machine to the next state), and a fourth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program that provides at least one function block output) that provides at least one function block output (see e.g., Fig. 14; i.e., the output corresponds to the output of node Done)for use in controlling one more field devices (see e.g., col. 5, lines 22 – 27; i.e., the target device corresponds to a field device), wherein the second, third, and fourth code is fixed (see e.g., col. 22 – col. 32; i.e., the code is used for determination of state transition, and current state, wherein the code is prewritten and included in the graphical program), but does not teach a user modifiable state machine configuration database. Klapper et al. teaches a matrix database stored on a memory (see e.g., para. [0028]; i.e., the) user modifiable database (see e.g., para. [0037]; i.e., the matrix databases parameters are user defined). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the computer readable medium, first code, second code, third code, and fourth code of Kodosky et al. with the user defined parameters of a matrix database of Klapper et al. because the matrix database is transferable to a matrix functional unit, an output response functional unit and an intersection functional unit, which configures the predefined input parameter logic, predefined output response logic, and predefined intersection logic (see e.g., para. [0008]; i.e., the database matrix has a matrix tool for inputting data into the database).

As to dependent claim 59, Kodosky et al teaches a first computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) with a first, second, third and forth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, and fourth code is incorporated into the graphical program) but does not teach a user modifiable state machine configuration database being stored on a computer readable medium. Klapper et al. teaches a "Cause and Effect Matrix" that is user definable (see e.g., para. [0037]; i.e., the matrix databases parameters are user defined) stored on a computer readable medium (see e.g., para. [0028]; i.e., the "Cause and Effect Matrix" is stored on memory element 14). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the computer readable medium, first code, second code, third code, and fourth code of Kodosky et al. with the user defined parameters of a matrix database of Klapper et al. because the matrix database is transferable to a matrix functional unit, an output response functional unit and an intersection functional unit, which configures the predefined input parameter logic, predefined output response logic, and predefined intersection logic (see e.g., para. [0008]; i.e., the database matrix has a matrix tool for inputting data into the database).

As to dependent claim 60, Kodosky et al. teaches a first computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) with a first, second, third and forth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, and fourth code is incorporated into the graphical program) but does not teach a user modifiable state machine configuration database being stored on a computer readable medium different from the first computer readable medium. Klapper et al. teaches a "Cause and Effect Matrix" that is user definable (see e.g., para. [0037]; i.e., the matrix databases parameters are user defined) that is stored on a second computer readable medium (see e.g., para. [0051]; i.e., the "Cause and Effect Matrix is storable on memory element 14 and programmable logic controllers (PLC) 78). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the computer readable medium storing a first, second, third, and forth code Kodosky et al. with the "Cause and Effect Matric" stored on a second computer readable medium of Klapper et al. because the matrix database is stored in a format that enables the user to upload the database to a computer for editing, modification, and comparison (see e.g., para. [0051]).

As to dependent claim 61, Kodosky et al. teaches a function block entity (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., the function block entity corresponds to the structured node function blocks A through Done), wherein one input comprises a plurality of inputs (see e.g., col. 14, lines 17 – 19; i.e., the Enable inputs comprises other inputs in order to execute at a proper time), and second code comprises a fifth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, and fifth code is incorporated into the graphical program) stored on a computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) to determine the next state (see e.g., col. 19, lines 48 – 59; i.e., the output of one node is the input for another node) based on priorities associated with the plurality of inputs (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., the alphanumeric letters corresponds to the priorities for sequential traversal of nodes, wherein each input must follow the alphanumeric priority).

As to dependent claim 62, Kodosky et al. teaches a fifth code comprising code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, and fifth code is incorporated into the graphical program) to determine the next state further based on an order (see e.g., Fig. 14; i.e., the alphanumeric nodes corresponds to the order and each node has a plurality of inputs) associated with the plurality of inputs (see e.g., Fig. 14).

As to dependent claim 63, Kodosky et al. teaches a function block entity (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., the function block entity corresponds to the structured node function blocks A through Done) comprising code (see e.g., col. 22 – col. 33; i.e., the code is included with the graphical program) stored on the first computer readable medium (see e.g., Fig. 14 and col. 20, lines 4 – 14; i.e., the code corresponds to the graphical program stored on a CD-ROM, magnetic media, or floppy disk that executes code to perform the next state determination of the state machine) to determine whether a state transition is to occur based on at least one input and state transition data (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., a state transition of a corresponding node will occur if the input satisfies the nodes condition, such as the output of node A as the input of node B), a second code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program) to determine the next state if a state transition data is to occur (see e.g., col. 19, lines 48 – 59; i.e., the loop determines the next state of a state transition), and a third code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program) to set the current state of the state machine to the next state if a state transition is to occur (see e.g., col. 20, lines 4 – 14; i.e., when node Done is reached, the current state of the state machine is set to the next state).

As to dependent claim 64, Kodosky et al. teaches the second code comprises the fifth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, and fifth code is incorporated into the graphical program) stored on the computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) to determine if any of the input is a particular value (see e.g., Fig. 14 and col. 19, lines 53 - 54; i.e., the particular value corresponds to the Loop Enable signal, wherein the state machine remains in state B until the signal is "true").

As to dependent claim 65, Kodosky et al. teaches the second code comprises a sixth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, fifth, and sixth code is incorporated into the graphical program) stored on the computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) to determine at least one input (see e.g., Fig. 14; i.e., the input corresponds to the arrow of node A pointing to node B, in which the output of node A is the input of node B) that a particular value that also corresponds to a state change based on the state transition data (see e.g., Fig. 14 and col. 19, lines 53 - 54; i.e., the particular value and state transition data corresponds to the Loop Enable signal, wherein the state machine remains in state B until the signal is "true").

As to dependent claim 66, Kodosky et al. teaches a function block entity (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., the function block entity corresponds to the structured node function blocks A through Done) comprising seventh code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, fifth, sixth and seventh code is incorporated into the graphical program) stored on a computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) to select one or more input (see e.g., Fig. 13 and col. 19, lines 36 – 47; i.e., the function block for a structured node corresponds to Fig. 14, wherein the function block for a specific node comprises multiple inputs, such as "Enable In input", "Period input", "Phase input", and "Loop Control input") that is a particular value (see e.g., col. 19, lines 48 – 59; i.e., the selection of inputs of a particular value corresponds to the input requirements of each node) that corresponds to a state change (see e.g., col. 19, lines 48 – 59; i.e., the state change corresponds to satisfying the requirements of each node and proceeding to the next node) if there is at least one input that is a particular value and that corresponds to a state change from the current state (see e.g., col. 19, lines 48 – 59; i.e., the particular value corresponds to the input requirements of each node, which when satisfied, causes a state change from the current state).

As to dependent claim 67, Kodosky et al. teaches the one input comprises a plurality of inputs (see e.g., col. 14, lines 14 – 19; i.e., the Enable input comprises other inputs such as master clock signal output and Done input), wherein the seventh code comprises an eighth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, fifth, sixth, seventh and eighth code is incorporated into the graphical program) stored on a computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) to select at least one input based on priorities associated with the plurality of inputs (see e.g., Fig. 14).

As to dependent claim 68, Kodosky et al. teaches the eighth code comprises code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, fifth, sixth, seventh and eighth code is incorporated into the graphical program) to select one of the inputs of the plurality of inputs (see e.g., col. 17, lines 39 – 50; i.e., the AND-gate is used to determine whether the inputs that are a particular value are selected and received) based on an order associated with the plurality of inputs (see e.g., Fig. 14; i.e., the alphanumeric nodes corresponds to the order and each node has a plurality of inputs).

As to dependent claim 69, Kodosky et al teaches a function block entity see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., the function block entity corresponds to the structured node function blocks A through Done) wherein the second code comprises a fifth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, and fifth code is incorporated into the graphical program) stored on the first computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) to determine one or more of the at least one input (see e.g., Fig. 14 and col. 19, lines 49 – 59; i.e., each node has a plurality of inputs, such as AND-gates, in which each AND-gate also has a plurality of inputs) that would cause a state change (see e.g., col. 20, lines 8 – 14; i.e., the state change corresponds to the successful traversal of the while loop), but does not teach determining a state change from a current state based on state transition data stored on a second computer readable medium. Klapper et al. teaches a second computer readable medium (see e.g., para. [0051]; i.e., the "Cause and Effect Matrix is storable on memory element 14 and programmable logic controllers (PLC) 78) for determining a state change based on the state transition data stored on the second computer readable medium (see e.g., para. [0050]; i.e., the determination of a state change from a current state due to state transition data stored on a second computer readable medium corresponds to the PLC 78 executing the logical signals for each cause and effect, which results in an intersection corresponding to a state change). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the function block entity of Kodosky et al. with the determination of state changes and second computer readable medium of Klapper et al. because by starting with a predefined and configurable functional unit for the cause and effect, the user may specifically configure each particular cause and effect of a matrix (see e.g., para. [0027]).

As to dependent claim 70, Kodosky et al. teaches a second code further comprising a sixth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, fifth, and sixth code is incorporated into the graphical program) stored on the first computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) to determine if one or more inputs would cause a state change and is also a particular value (see e.g., Fig. 14 and col. 19, lines 53 - 54; i.e., the particular value corresponds the Loop Enable signal, wherein the state machine remains in state B until the signal is "true").

As to dependent claim 71, Kodosky et al. teaches a fourth code comprising a fifth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, and fifth code is incorporated into the graphical program) stored on a computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) to provide a plurality of function block outputs (see e.g., Fig. 14 and col. 19, lines 48 – 59; i.e., the plurality of output function block outputs corresponds to Set Enable out signal, and Clear Output).

As to dependent claim 72, Kodosky et al. teaches a fifth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, and fifth code is incorporated into the graphical program) comprising code (see e.g., col. 22 – col. 33; i.e., the code provided is used to execute embodiments of Fig. 14) to provide each of at least some of the plurality of function block outputs (see e.g., Fig. 14; i.e., the function block outputs correspond to the arrows drawn from node to another) that are indicative whether the current state of the state machine corresponds to a respective plurality of possible states (see e.g., col. 19, lines 60 – 67; i.e., the structured node function block outputs are evaluated within the loop, and if the period is done and the loop has not yet completed, then the state machine proceeds to state Error) of the state machine (state machine – see e.g., col. 19, line 66).

As to dependent claim 73, Kodosky et al. teaches displaying (see e.g., col. 2, lines 58 – 67) on a display device (video display subsystem 180 – see e.g., Fig. 2 and col. 10, lines 1 – 6), state transition of function blocks and state machines (see e.g., col. 19, lines 48 – 67) through a graphical user interface (graphical program – see e.g., col. 2, lines 58 – 67), providing a plurality of function block outputs (see e.g., Fig. 14 and col. 19, lines 48 – 59), and the fifth code comprising a sixth and seventh code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, fifth, sixth and seventh code is incorporated into the graphical program), but does not teach retrieving, based on the current state, data indicative of appropriate values for at least some of the plurality of state machine function block outputs from a user configurable output configuration database, and setting at least some of the plurality of function block outputs to the appropriate value. Klapper et al. teaches retrieving data indicative of appropriate values for at least some of the plurality of state machine function block outputs (see e.g., para. [0045]; i.e., Effect Latch Function logic element 124, receives a Restart Tag, and a Max Restart Time value 126 from the effect database) from a user configurable (see e.g., para. [0037]) output configuration database (see e.g., para. [0045]; i.e., the output configuration database corresponds to the effect database), and setting at least some of the plurality of function block outputs to the appropriate value (see e.g., para. [0047]). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the graphical user interface, state machine, function block, and providing a plurality of function block output of Kodosky et al. with the retrieval of function block outputs from a output configuration table, and setting the plurality of function block outputs to the appropriate value of Klapper et al. because the logic that implements the matrix is automatically defined and established by transferring the matrix to the programmable logic controllers (see e.g., para. [0049]).

As to dependent claim 74, Kodosky et al teaches displaying (see e.g., col. 2, lines 58 – 67) on a display device (video display subsystem 180 – see e.g., Fig. 2 and col. 10, lines 1 – 6), state transition of function blocks and state machines (see e.g., col. 19, lines 48 – 67) through a graphical user interface (graphical program – see e.g., col. 2, lines 58 – 67), providing a plurality of function block outputs (see e.g., Fig. 14 and col. 19, lines 48 – 59), and a plurality of codes, but does not teach a state machine configuration database and an output configuration database that are stored on the same computer readable medium. Klapper et al. teaches a state machine configuration database (see e.g., para. [0037]; i.e., the matrix databases parameters are user defined) and an output configuration database (see e.g., para. [0045]; i.e., the output configuration database corresponds to the effect database) that are stored on the same computer readable medium (memory 14 – see e.g., para. [0028]; i.e., the state machine and output configuration database corresponds to the "Cause and Effect Matrix" that is stored on memory 14 of the system). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the state transition of function blocks and state machines through the graphical user interface of Kodosky et al. with the "Cause and Effect Matrix" that is stored on the same computer readable medium of Klapper et al. because the program code means can be accessed by a general computer or a special purpose computer (see e.g., para. [0054]).

As to dependent claim 75, Kodosky et al. teaches a first computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) with a first, second, third, forth, fifth, sixth, and seventh code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, and fourth code is incorporated into the graphical program) but does not teach a user modifiable state machine configuration database and an output configuration database being stored different computer readable medium. Klapper et al. teaches user configuration database that is user definable (see e.g., para. [0037]; i.e., the matrix databases parameters are user defined) that is stored on a computer readable medium (see e.g., para. [0051]; i.e., the "Cause and Effect Matrix is storable on memory element 14) and an output configuration database that is stored on a different computer readable medium (see e.g., para. [0026]; i.e., once the user defines the matrix, the data can be transferred to a programmable logic controller (PLC) 78, which corresponds to a separate computer readable medium). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the computer readable medium storing code of Kodosky et al. with the state machine configuration database and a output configuration table stored on a separate computer readable medium of Klapper et al.

because the matrix database is stored in a format that enables the user to upload the database to a computer for editing, modification, and comparison (see e.g., para. [0051]).

As to dependent claim 76, Kodosky et al. teaches a fifth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, and fifth code is incorporated into the graphical program) stored on the first computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks), wherein receiving an input indicative of whether the state machine function block is to be disabled (see e.g., col. 19, lines 66 – 67; i.e., the error input corresponds to indicating whether the state machine function block is to be disabled); and a sixth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, fifth, and sixth code is incorporated into the graphical program) stored on the first computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) if the input indicative of whether the state machine function block is to be disabled indicates that the state machine function block is to be disabled, setting the current state of the state machine to a disabled state (see e.g., Fig. 14 and col. 19, lines 66 – 67; i.e., once state Error is reached, the loop completes, therefore the current state of the state machine is set to a disabled state).

As to dependent claim 78, Kodosky et al. teaches at least one data input comprises at least one signal (see e.g., col. 14, lines 14 – 19; i.e., the data inputs and signals corresponds to Enable input, Clear_Enable input, and master clock signal output) associated with at least one of a process control system (industrial automation system 140 – see e.g., col. 8, line 5 – 18; i.e., the industrial automation system 140 is used to construct the input/output of function blocks), a simulation of a process control system (see e.g., col. 7, lines 25 – 26; i.e., analyzing the graphical program is a form of simulation), a safety system, and a simulation of a safety system.

7. **Claims 12 – 14, 28 – 30, and 52 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kodosky et al. (Patent No. 6,954,724) in view of Khrapunovich et al. (Patent No. 5,845,063).**

As to dependent claims 12 and 29, Kodosky et al. teaches a computer implemented method and corresponding apparatus, wherein the steps/means for a computing device (computer 102 – see e.g., Fig. 2 and col. 7, lines 19 – 26) and a graphical user interface (graphical program – see e.g., col. 2, lines 58 – 67) for configuring how a state machine (state machine – see e.g., col. 19, lines 49 – 59; i.e., the while loop function block is used to implement the transition of the state machine) is to transition among a plurality of states (see e.g., Fig. 16 and col. 48 – 67; i.e., the while loop function block controls the state machines transition of states), but does not teach receiving data indicative of how to handle inputs that have a BAD status and storing the data indicative of how to handle inputs that have a BAD status. Khrapunovich et al. teaches receiving data (see e.g., col. 4, lines 65 – 67; i.e., the data corresponds to rules embedded in the function block) indicative of how to handle inputs (see e.g., Fig. 3) that have a BAD status (INPUT 1 – see e.g., Fig. 3 and col. 4, lines 48 – 64; i.e., INPUT 1 of Fig. 3 is a signal that corresponds to a BAD status), and storing the data indicative of how to handle inputs that have BAD status (see e.g., col. 4, lines 48 – 67; i.e., embedding corresponds to the storing of data indicative of how to handle inputs with BAD status). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made in to incorporate the computing device and graphical user interface of Kodosky et al. with the data indicating how to handle inputs with BAD signals of Khrapunovich et al. because the data handling the BAD status input prevents the latching of the closed loop (see e.g., col. 4, lines 65 – 67 and col. 5, lines 1 – 5).

As to dependent claim 13 and 28, Kodosky et al. teaches a computer implemented method and corresponding apparatus, wherein the steps/means for a computing device (computer 102 – see e.g., Fig. 2 and col. 7, lines 19 – 26) and a graphical user interface (graphical program – see e.g., col. 2, lines 58 – 67) for configuring how a state machine (state machine – see e.g., col. 19, lines 49 – 59; i.e., the while loop function block is used to implement the transition of the state machine) to transition among a plurality of states (see e.g., Fig. 16 and col. 48 – 67; i.e., the while loop function block controls the state machines transition of states), with at least one input comprising a plurality of inputs (see e.g., Fig. 17 and col. 14, lines 41 – 45) but does not teach receiving data indicative of priorities associated with the plurality of inputs, and storing data indicative of how to handle inputs that have BAD status. Khrapunovich et al. teaches receiving data indicative of priorities (see e.g., Fig. 1 – Fig.3; i.e., the arrows indicated priority and the direction of the input from left to right and top to bottom) associated with the plurality of inputs (see e.g., Fig. 1 – Fig. 3; i.e., INPUT 1 and INPUT 2 corresponds to the plurality of inputs), and storing data indicative of how to handle inputs that have BAD status (see e.g., col. 4, lines 48 – 67; i.e., embedding corresponds to the storing of data indicative of how to handle inputs with BAD status). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the graphical user interface and one input comprising a plurality of inputs of Kodosky et al. with the data indicative of whether to ignore at least one input and priorities of Khrapunovich et al. because the data indicating priority allows the user to visualize the traversal of inputs and the data indicative of handling the BAD status input prevents the latching of the closed loop (see e.g., col. 4, lines 65 – 67 and col. 5, lines 1 – 5).

As to dependent claims 14 and 30, Kodosky et al. teaches a computer implemented method and corresponding apparatus, wherein the steps/means for a computing device (computer 102 – see e.g., Fig. 2 and col. 7, lines 19 – 26) and a graphical user interface (graphical program – see e.g., col. 2, lines 58 – 67) for configuring how a state machine (state machine – see e.g., col. 19, lines 49 – 59; i.e., the while loop function block is used to implement the transition of the state machine) is to transition among a plurality of states (see e.g., Fig. 16 and col. 48 – 67; i.e., the while loop function block controls the state machines transition of states), but does not teach receiving data indicative of ignoring at least one input by the state machine, and storing the data indicative of whether at least one input should be ignored by the state machine. Khrapunovich et al. teaches data indicative (see e.g., col. 4, lines 41 – 67; i.e., the data corresponds to the options within the function block) of whether at least one input should be ignored (see e.g., col. 4, lines 41

– 67), and storing the data indicative of whether to ignore at least one of the inputs (see e.g., col. 4, lines 65 – 67). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the computing device and graphical user interface of Kodosky et al. with the data storing and indication of whether to ignore an input of Khrapunovich et al. because the Force Status Function block forces the status bits to zero in order to prevent propagation of unwanted status bits in a closed loop (see e.g., col. 4, lines 39 – 67 and col. 5, lines 1 – 5).

As to dependent claims 51 and 52, Kodosky et al. teaches a computing device (computer 102 – see e.g., Fig. 2 and col. 7, lines 19 – 26) and a graphical user interface (graphical program – see e.g., col. 2, lines 58 – 67) for configuring how a state machine (state machine – see e.g., col. 19, lines 49 – 59; i.e., the while loop function block is used to implement the transition of the state machine) to transition among a plurality of states (see e.g., Fig. 16 and col. 48 – 67; i.e., the while loop function block controls the state machines transition of states), with at least one input comprising a plurality of inputs (see e.g., Fig. 17 and col. 14, lines 41 – 45) but does not teach receiving an input indicative of whether the state machine function block is forced to an initial state, setting the current state of the state machine to the initial state, and the input indicative of whether the function block is to be enabled or disabled is a single input. Khrapunovich et al. teaches an instance where it would be desirable to force a signal to an initial state (see e.g., col. 4, lines 36 – 38; i.e., if the inputs are faulty, it would be desirable to reset all the signals), setting the current state of the state machine to the initial state (see e.g., col. 5, lines 4 – 5; i.e., setting the current state to the initial state corresponds to setting all the status bits to zero), and the single input indicative of whether the function block is to be enabled or disabled (see e.g., col. 4, lines 33 – 47). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the computing device and graphical user interface of Kodosky et al. with the setting and forcing a function block to an initial state of Khrapunovich et al. because Force Status Function block forces the status bits to zero in order to prevent propagation of unwanted status bits in a closed loop (see e.g., col. 4, lines 39 – 67 and col. 5, lines 1 – 5).

8. **Claims 6 – 8, 23 – 25, 77 and 79 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kodosky et al. (Patent No. 6,954,724) in view of Klapper et al. (Pub No. 2002/0194218) and further in view of Khrapunovich et al. (Patent No. 5,845,063).**

As to dependent claims 6 and 23, Kodosky et al. teaches a computer implemented method and corresponding apparatus, wherein the steps/means for a computing device (computer 102 – see e.g., Fig. 2 and col. 7, lines 19 – 26), function blocks (function block – see e.g., col. 11, lines 59 – 67), state machines (state machine – see e.g., col. 19, lines 49 – 59), a plurality of inputs (see e.g., col. 8, lines 5 – 18), state transition (see e.g., col. 13, lines 40 – 54), and a plurality of graphical elements (see e.g., Fig. 16 and col. 20, lines 44). Klapper et al. teaches a plurality of cells associated with possible parings (see e.g., para. [0029]), via an input device (see e.g., para. [0029]), of graphical elements with at least one input that is indicative of a next state (see e.g., para. [0027). Both Kodosky et al. and Klapper et al. do not teach the value being a logical one, logical zero, a logical TRUE value, and a logical FALSE value. Khrapunovich et al. teaches input signals and bits being in the form of zero and ones (see e.g., Microsoft Computer Dictionary 5th Edition and col. 4, lines 41 – 47; i.e., binary is defined as "values are expressed as combinations of two digits, 0 and 1.The two digits can represent the logical values *true* and *false*…"). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the computing device and graphical interface of Kodosky et al. as modified by the "Cause and Effect Matrix" of Klapper et al. with the logical zero and logical one of Khrapunovich et al. because this allows the function block to determine if the block propagates or ignores the status of the input signal (see e.g., col. 4, lines 41 – 47; i.e., the logical zero and logical one are indicators of propagation or closed loop status).

As to dependent claim 7 and 24, Kodosky et al. teaches a computer implemented method and corresponding apparatus, wherein the steps/means for a computing device (computer 102 – see e.g., Fig. 2 and col. 7, lines 19 – 26), function blocks (function block – see e.g., col. 11, lines

59 – 67), state machines (state machine – see e.g., col. 19, lines 49 – 59), a plurality of inputs (see e.g., col. 8, lines 5 – 18), state transition (see e.g., col. 13, lines 40 – 54), and input device (keyboard – see e.g., col. 7, lines 19 – 26), and a plurality of graphical elements (see e.g., Fig. 16 and col. 20, lines 44). Khrapunovich et al. teaches input signals and bits being in the form of zero and ones (see e.g., Microsoft Computer Dictionary 5th Edition and col. 4, lines 41 – 47; i.e., binary is defined as "values are expressed as combinations of two digits, 0 and 1.The two digits can represent the logical values *true* and *false*..."). Both Kodosky et al. and Khrapunovich et al. do not teach receiving data, via an input device, indicative of a number of inputs in the at least one input, and determining the number of cells in the first plurality of cells based on the number of inputs. Klapper et al. teaches a "Configure Tab" (see e.g., para. [0029]) for an input (see e.g., Fig. 4 and para. [0029]; i.e., the input corresponds to "High Furnace Pressure" in "Desc" input tag) with a plurality of inputs (see e.g., Fig. 4 and para. [0029]; i.e., the plurality of inputs corresponds "number of inputs" input 42, which allows the user to define the number of sensors the cause will monitor), and the determining the number of cells in the first plurality of cells based on the number of inputs (see e.g., Fig. 1, Fig. 4 and para. [0029]; i.e., in this example, the user has chosen 3 inputs as illustrated in "number of inputs" 42, which corresponds to "Tag1, Tag2, and Tag3" of Fig. 4, and is further displayed on the matrix of Fig. 1). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the computing device, input device, and graphical interface of Kodosky et al. as modified by the inputs and bits being in the form of zeros and ones of Khrapunovich et al. with the "Configure Tab" of Klapper et al. because the matrix programming tool allows a user to define matrix parameters (see e.g., para. [0029]).

As to dependent claim 8 and 25, Kodosky et al teaches a computer implemented method and corresponding apparatus, wherein the steps/means for a computing device (computer 102 – see e.g., Fig. 2 and col. 7, lines 19 – 26), function blocks (function block – see e.g., col. 11, lines 59 – 67), state machines (state machine – see e.g., col. 19, lines 49 – 59), a plurality of inputs (see e.g., col. 8, lines 5 – 18), state transition (see e.g., col. 13, lines 40 – 54), and input device (keyboard – see e.g., col. 7, lines 19 – 26), and a plurality of graphical elements (see e.g., Fig. 16

and col. 20, lines 44). Khrapunovich et al. teaches input signals and bits being in the form of zero and ones (see e.g., Microsoft Computer Dictionary 5th Edition and col. 4, lines 41 – 47; i.e., binary is defined as "values are expressed as combinations of two digits, 0 and 1.The two digits can represent the logical values *true* and *false*..."). Both Kodosky et al. and Khrapunovich et al. do not teach receiving data, via an input device, indicative of a number of inputs in the at least one input, determining the number of cells in the first plurality of cells based on the number of inputs, and determining the number of cells in the first plurality of cells based on the number of states. Klapper et al. teaches a "Configure Tab" (see e.g., para. [0029]) for an input (see e.g., Fig. 4 and para. [0029]; i.e., the input corresponds to "High Furnace Pressure" in "Desc" input tag) with a plurality of inputs (see e.g., Fig. 4 and para. [0029]; i.e., the plurality of inputs corresponds "number of inputs" input 42, which allows the user to define the number of sensors the cause will monitor), the determining the number of cells in the first plurality of cells based on the number of inputs (see e.g., Fig. 1, Fig. 4 and para. [0029]; i.e., in this example, the user has chosen 3 inputs as illustrated in "number of inputs" 42, which is further displayed on the matrix of Fig. 1), and determining the number of cells with respect to the number of states (see e.g., Fig. 4 and para. [0029]; i.e., the number of states corresponds to "Tag1, Tag2, and Tag3" of Fig. 4, and is further displayed on the matrix of Fig. 1). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the computing device, input device, and graphical interface of Kodosky et al. as modified by the inputs and bits being in the form of zeros and ones of Khrapunovich et al. with the "Configure Tab" of Klapper et al. because the matrix programming tool allows a user to define matrix parameters (see e.g., para. [0029]).

As to dependent claim 77, Kodosky et al teaches a first computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks) storing a fifth and sixth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program, wherein the first, second, third, fourth, fifth, and sixth code is incorporated into the graphical program), and setting the current state of the state machine to an initial state (see e.g., col. 20, lines 8 – 14; i.e., the current state of the state machine are set to an initial state once the while loop has successfully traversed the structured node function blocks). teaches a matrix database stored on a memory (see e.g., para. [0028]; i.e., the) user modifiable database (see e.g., para. [0037]; i.e., the matrix databases parameters are user defined). Both Kodosky et al. and Klapper et al. do not teach receiving input indicative of whether the state machine function block is to be forced to an initial state. Khrapunovich et al. teaches an instance where it would be desirable to force a signal to an initial state (see e.g., col. 4, lines 36 – 38; i.e., if the inputs are faulty, it would be desirable to reset all the signals), and setting the current state of the state machine to the initial state (see e.g., col. 5, lines 4 – 5; i.e., setting the current state to the initial state corresponds to setting all the status bits to zero).). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the computing device, graphical user interface, fifth and sixth code stored on a computer readable medium of Kodosky et al. as modified by the "Cause and Effect Matrix" of Klapper et al. with the setting and forcing a function block and state machine to an initial state of Khrapunovich et al. because Force Status Function block forces the status bits to zero in order to prevent propagation of unwanted status bits in a closed loop (see e.g., col. 4, lines 39 – 67 and col. 5, lines 1 – 5).

As to dependent claim 79, Kodosky et al teaches a first computer readable medium (see e.g., col. 8, lines 53 – 57; i.e., the computer readable medium corresponds to a CD-ROM, magnetic media, or floppy disks), a first code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program) to receive input (see e.g., col. 13, lines 21 –39; i.e., the inputs corresponds to the user manipulating icons on the graphical editor) corresponding to a process plant (see e.g., col. 8, lines 5 – 18), second code to determine the next state of the state machine (see e.g., Fig. 14 and col. 20, lines 4 – 14; i.e., the code corresponds to the graphical program stored on a CD-ROM, magnetic media, or floppy disk that executes code to perform the next state determination of the state machine), a third code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program that includes setting the state machine to the next state) to set the current state of the state machine to the next state (see e.g., col. 20, lines 8 – 14; i.e., successful traversal to node Done sets the state machine to the next state), and a fourth code (see e.g., col. 8, lines 55 – 57; i.e., the code corresponds to the graphical program that provides at least one function block output) to provide one function block output (see e.g., Fig. 14; i.e., the output corresponds to the output of node Done) in controlling the field device (see e.g., col. 5, lines 22 – 27; i.e., the target device corresponds to a field device). Klapper et al. teaches a matrix database stored on a memory (see e.g., para. [0028]; i.e., the) user modifiable database (see e.g., para. [0037]; i.e., the matrix databases parameters are user defined). Both Kodosky et al. and Klapper et al. do not teach the masking capability of at least one input. Khrapunovich et al. teaches masking (see e.g., Microsoft Computer Dictionary 5th Edition and col. 4, lines 41 – 57; i.e., the definition of masking is "a binary value used to selectively screen out or let through certain bits in a data value") capabilities of at least one input (see e.g., col. 4, lines 41 – 57; i.e., the function block includes various options such as allowing propagation or ignoring the status of an input by forcing the binary bits to a zero or one). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the computing device, graphical interface, first, second, third, and fourth code of Kodosky et al. as modified by the "Cause and Effect Matrix" of Klapper et al. with the masking of inputs of Khrapunovich et al. because this allows the function block to determine if the block propagates or ignores the status of the input signal (see e.g., col. 4, lines 41 – 47; i.e., the logical zero and logical one are indicators of propagation or closed loop status).

9. **Claims 10, 11 and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kodosky et al. (Patent No. 6,954,724) in view of Klapper et al. (Pub No. 2002/0194218) and further in view of Larson et al. (Patent No. 6,369,836).**

As to dependent claim 10 and 27, Kodosy et al. teaches a computer implemented method and corresponding apparatus, wherein the steps/means for a computing device (computer 102 – see e.g., Fig. 2 and col. 7, lines 19 – 26) with a plurality of graphical elements (see e.g., Fig. 16 and col. 20, lines 44; i.e., the graphical elements corresponds to 33"Adder 532", "AND gate 534" etc.), function blocks (function block – see e.g., col. 11, lines 59 – 67) with a plurality of outputs (see e.g., Fig. 19) of states of the state machine (state machine – see e.g., col. 19, lines 49 – 59; i.e., the while loop function block is used to implement the transition of the state machine). Klapper et al. teaches a plurality of cells associated with a function block (see e.g., Fig. 1 and para. [0027]; i.e., the indication of state transition can be viewed through the intersection of Cause row and Effect column). Both Kodosky et al. and Klapper et al. both do not teach a second plurality of cells associated with function blocks, wherein each cell of the second plurality of cells corresponds to a plurality of outputs of the function block, and receiving output configuration data associated with the second plurality of cells via an input device, wherein the data is indicative of an output value, and storing the output configuration data on a second computer readable medium. Larson et al. teaches a second plurality of cells (variable detail pane 28 – see e.g., col. 12, lines 5 – 14) associated with the function block (see e.g., col. 12, lines 5 – 14; i.e., the first pane 27, second pane 28, and third pane 29 as a whole correspond to a function block) wherein each cell of the second plurality of cells corresponds to a plurality of outputs (see e.g., Fig. 27; i.e., second pane 28 consists of two VAR_OUTPUT under column var type), and receiving output configuration data associated with the second pluraliy of cells (see e.g., col. 12, lines 35 – 36) via an input device (keyboard input 3B – see e.g., col. 3, line 54). Larson et al. further teaches some of the second plurality of cells includes data indicative of an output value (see e.g., col. 12, lines 31 – 36; i.e., second pane 28 includes output data, which correspond to VAR_OUT), and storing (see e.g., col. 11, line 31) the output configuration data on a computer readable medium (see e.g., col. 11, lines 28 – 33). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the graphical elements, and function blocks, state transition of state machines of Kodosky et al. as modified by the "Cause and Effect Matrix" of Klapper et al. with the second plurality of cells associated with function blocks of Larson et al. because the second plurality of cells is used for displaying any variables associated with the cause, effect, or intersection selected by the user (see e.g., col. 11, lines 8 – 14; i.e., the display of variables in the second pane 28 allows easier visualization of cause, effect, or intersection attributes).

As to claim 11, Kodosky et al. teaches a first computer readable medium (see e.g., col. 13, lines 40 – 54; i.e., the arrangement of graphical elements on the screen by the user is stored by the graphical programming system) comprises the second computer readable medium (see e.g., col. 8, lines 61 – 64; i.e., the computer system comprises and stores a software program on a hard drive and memory).

It is noted that any citation to specific pages, columns, lines, or figures in the prior art references and any interpretation of the references should not be considered to be limiting in any way. A reference is relevant for all it contains and may be relied upon for all that it would have reasonably suggested to one having ordinary skill in the art. In re *Heck*, 699 F.2d 1331, 1332-33,216 USPQ 1038, 1039 (Fed. Cir. 1983) (quoting In re *Lemelson*, 397 F.2d 1006,1009, 158 USPQ 275, 277 (CCPA 1968)).

Response to Arguments

10. Applicant's arguments filed 03/19/2007 have been fully considered but they are not persuasive.

Applicant's argument that Kodosky does not teach every element of claim 1

Applicant argues that Kodosky does not teach every element of claims 1 because the applicant interprets the prior art as not teaching the limitation of "providing a graphical user interface via a display device for configuring , at least in part, how the state machine is to transition among a plurality of states, wherein the graphical user interface includes a plurality of graphical elements, wherein at least some of the graphical elements can be used to indicate desired transitions between states". The applicant also argues that the Office refers to specific sections of the reference that do not teach the limitations of the claim (See Page 21, Para 3).

The Examiner disagrees.

MPEP 2123 and 2144.01 states that a reference is available for all that is relevant and suggested to one of ordinary skill in the art and not just the cited sections. In this case, the examiner provides the following citations to address claim 1:

In the preamble:

A Device – 102 – Fig 2, Col 7, lines 19-26

A Display – 180 Fig. 2, Col. 10 lines 1-6

Process Plant – Col 11, Lines 59-67

Associated with a Function Block, Col 2, lines 8-15 and Col 19, lines 49-59

State Machine – Col 19, Lines 49-59

Graphical Program – Col 2, Lines 58-67

Shows how state transition – Fig 16 and Col 48 – 67

Plurality of Elements - Fig. 16 + Col. 20 line 44

Transition between States – Fig. 16, Col. 20 lines 30-59

At least One input with process plant – Col 8, lines 5-18

Storing state transition data – Col. 13, lines 40-54

While the cited sections refer to subject matter used to reject the claim, the cited sections have features of the same embodiment that are also mentioned in other parts of the reference. Therefore, in the rejection above the Examiner refers to a display as shown in figure 2, 180 that is part of section 102 of Figure 1. In the description of Figure 1, column 9, Lines 1-45 teaches that the system incorporates a graphical programming system with an interface. Kodosky also teaches that the graphical programming environment results in the graphical display interface (panel) being displayed to the user (See column 11, lines 32-57). Kodosky teaches a process of allowing the user to configure the graphical program by manipulating features within the interface (See Figure 5 and column 13, lines 20-55). Further, the graphical programming environment and the LabView program along with the manuals are incorporated by reference (See column 7, lines 10-18). The system of Kodosky therefore provides that the interface gives the user the ability to construct a program that manages state of a FPGA that controls an instrument. The FPGA has a series of gates that incorporate states of the machine where inputs to the gates control the output of the machine. With a given input the state of the machine could be on, off, operating, etc. The Elements on the display will reflect in real time the status of the machine as data paths emanating from the nodes in the graphical program (See column 14, lines 14-41). Further, Kodosky states that a portion of the program is converted to a hardware implementation and is executed locally in the CPU while the remaining section of the program is in the interface allowing the user to make changes based on the observed state of the machine (See column 10, lines 39-67). Therefore, the Office has provided a specific description of how Kodosky teaches the limitations of the claims.

Applicant's argument that Kodosky in view of Klapper does not teach a state machine database that contains data where the states indicate the next state to where the machine should transition

Applicant argues that Kodosky In view of Klapper does not teach the limitations of claim 58 where "for each of at least some of the pairings of each of at least some of the plurality of states and each of the at least some of at least one input to the function block, indicative of a next state to which the state machine should transition when the state machine is in the corresponding state and when the corresponding input is a particular value" because the applicants interpret Klapper as not teaching data in the columns and rows that correspond to a state machine (See arguments page 24).

The Examiner disagrees.

In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986). In this case, the Office applied Kodosky in view of Klapper to teach the limitations of claim 58. The Office relied on Klapper to teach the single limitation of a modifiable state transition database where the parameters are user defined. Kodosky teaches that the inputs to the gate array are states to be input to a device. Klapper teaches the a specific logic array is implemented and automatically instantiated with a given input that effects the functional unit (See Para 0027). Kodosky teaches sending inputs to control a functional unit and Klapper teaches storing inputs sent to a functional unit for the purposes of creating a record of the effect of providing a specific input. Therefore, Klapper is a specific example of a process of verifying logic implementations of data sent to the field programmable devices of Kodosky and storing the information in the database so that users can see the effects of sending specific inputs to a particular node.

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Prior art Patent No. 6,442,512 can be applicable and pertinent to applicant's disclosure. Prior art disclosed by Sengupta et al. teaches a graphical user interface to build and specify flow sheet configurations, such as a process plant or refinery, in which dragging and dropping process units and related process plant icons on a graphical editor.

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Prior art Patent No. 5,838,563 can be applicable and pertinent to applicant's disclosure. Prior art disclosed by Dove et al. teaches a graphical user interface that allows dragging and dropping of process control environment icons on a pallet to create state transitions of function blocks and state machines. Dove et al. further teaches creating a process control function and connecting various control functions to produce state transition.

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Prior art Patent No. 6,931,288 can be applicable and pertinent to applicant's disclosure. Prior art disclosed by Lee et al. teaches editing function block diagrams through a user interface, in which each function block can be wired or connected using the interface.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Steven B. Theriault whose telephone number is (571) 272-5867. The examiner can normally be reached on M, W, F 10:00AM - 8:00 PM.
If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Weilun Lo can be reached on (571) 272-4847. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

SBT

[illegible]

WEILUN LO
SUPERVISORY PATENT EXAMINER